Efficient Algorithms for Finding Disjoint Paths in Grids * (Extended Abstract)

Wun-Tat Chan † Francis Y.L. Chin †

Abstract

The *reconfiguration problem* on VLSI/WSI processor arrays in the presence of faulty processors can be stated as the following integral multi-source routing problem [16]: Given a set of N nodes (faulty processors or sources) in an $m \times n$ rectangular grid where $m, n \leqslant N$, the problem to be solved is to connect the N nodes to distinct nodes at the grid boundary using a set of "disjoint" paths. This problem can be referred to as an *escape problem* [5] which can be solved trivially in $O(mnN)$ time.

By exploiting all the properties of the network, planarity and regularity of a grid, integral flow, and unit capacity source/sink/flow, we can optimally compress the size of the grid from $O(mn)$ to $O(\sqrt{mnN})$ and solve the problem in $O(d\sqrt{mnN})$, where d is the maximum number of disjoint paths found, for both the *edge-disjoint* and *vertex-disjoint* cases. In the worst case, d, m, n are $O(N)$ and the result is $O(N^{2.5})$. Note that this routing problem can also be solved with the same time complexity even if the disjoint paths have to be ended at another set of N nodes (sinks) in the grid instead of the grid boundary.

1 Introduction.

The combinatorial problem to be discussed in this paper is motivated by the problem of developing efficient algorithms for reconfiguring VLSI/WSI processor arrays in the presence of faulty processors [4, 15, 16]. This is called *reconfiguration problem* and can be stated as the following discrete multi-source routing problem [16]:

> Given a set of N nodes (faulty processors or sources) in an $m \times n$ rectangular grid, the problem to be solved is to connect the N nodes to distinct nodes (sinks) at the grid boundary using a set of "disjoint" paths (compensation paths).

If non-faulty processors are allowed to reconfigure along the grid lines by "single-track" switches, then the disjoint (compensation) path connecting a faulty processor to a non-faulty boundary processor should contain no faulty processors, and be limited to vertical or horizontal straight lines. An $O(N^2)$ algorithm was first proposed in [16], and later improved to $O(N \log N)$ in [1], to find a set of non-intersecting straight lines if a solution to this reconfiguration problem exists; otherwise the algorithm would provide no solution. The optimization problem of finding the maximum number of nodes that can be connected to the boundary by non-intersecting straight lines was studied and an $O(N^3)$ algorithm was first presented in [3], and later improved to $O(N^2 \log N)$ in [12].

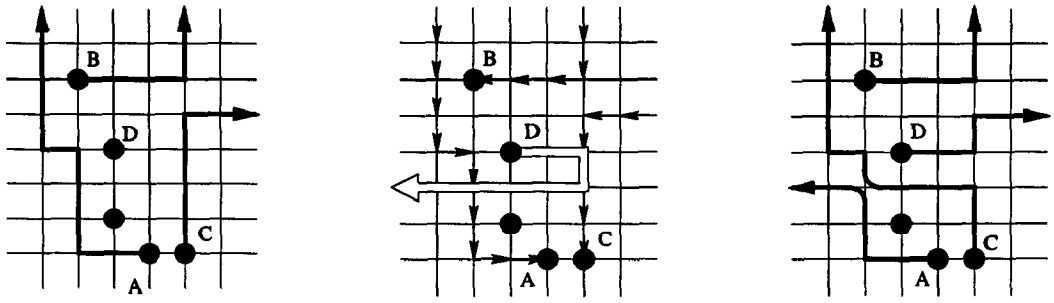
If the array of processors on a rectangular grid are equipped with "multi-track" switches, then the compensation paths would not be limited to straight lines. There are two interpretations of "disjoint" paths, *edge-disjoint* paths or *vertex-disjoint* paths. A set of paths are called *edge (vertex) disjoint* if no edge (vertex) is shared by more than one path. For the fault-tolerant reconfiguration problem, vertex-disjoint paths are required. This problem has been mentioned in [16] and was referred to as the *escape problem* in [5]. In this paper, both forms of problems, finding vertex-disjoint and edge-disjoint paths, are studied.

This reconfiguration problem can be viewed as a vertex-disjoint or edge-disjoint routing problem in a planar graph [8, 9, 11, 14, 17]. Most of these papers are concerned with the problem of finding disjoint paths where each path connects two specified vertices in the graph. A more relevant result is the optimal linear-time algorithm [14], an improvement on the $O(p \log p)$ result [18] to find the maximum number of vertex-disjoint (s,t)-paths for an $O(p)$-size planar graph. Unfortunately, the techniques used in these two algorithms cannot be applied to our problem of improving the time complexity in finding vertex-disjoint paths for the source nodes.

The edge-disjoint paths problem can be solved by reducing it to a multiple-source, multiple-sink flow problem [10] on a grid network, where all the N nodes are *sources* with unit supply, all the boundary vertices in the grid are *sinks* with unit demand, and every grid edge has unit capacity. As the grid is planar, it can be shown [10] that this multiple-source, multiple-sink problem on an $O(p)$-size planar network can be solved in $O(p^{4/3} \log p)$ time (or $O(N^{8/3} \log N)$ time for the reconfiguration problem if m, n are $O(N)$) by using fast shortest-path algorithm for planar graphs [7].

*The research is partially supported by an RGC grant 338/065/0022.

†Department of Computer Science, The University of Hong Kong. *E-mail:{wtchan, chin}@cs.hku.hk*



(a) A set of 3 disjoint paths.

(b) An augmenting path in auxiliary grid.

(c) A set of 4 disjoint path.

Figure 1: Example of finding of an augmenting path in auxiliary grid.

However, this algorithm [10] can only find a feasible solution if it exists and cannot find a solution with the maximum number of edge-disjoint paths. Alternatively, a multiple-source, multiple-sink problem can be reduced to a single-source, single-sink MAX-FLOW problem by connecting the sources to a super-source and sinks to a super-sink. However, this reduction may destroy the planarity of the graph. Solving the edge-disjoint paths problem as a unit-capacity MAX-FLOW problem [13] provides $O(p^{5/3})$ time complexity. The vertex-disjoint paths problem can be solved by "splitting" a vertex into two and connecting them by an edge with unit capacity which ensures that only one path can traverse a vertex [2]. This reduction also destroys the planarity property of the grid but results in a simple network [13] with unit capacity which leads to an $O(p^{1.5})$ (or $O(N^3)$ for the reconfiguration problem if m, n are $O(N)$) time algorithm for the vertex-disjoint routing problem.

In fact, the $O(mnd)$ time bound where d is the maximum number of disjoint paths, i.e., $O(N^3)$ time bound for an $O(N) \times O(N)$ grid with N source nodes, can also be achieved easily with the standard FORD-FULKERSON algorithm [6], by connecting the sources to a super-source and sinks to a super-sink and finding the augmenting paths from the super-source to the super-sink in the network. In the ith iteration, the existence of an augmenting path guarantees the existence of a set of i edge-disjoint paths from i source nodes to i boundary vertices. An *auxiliary grid* in the ith iteration is the *residual network* whose edges are with unit capacity. Alternatively, assume there are i disjoint paths from the super-source to super-sink. The auxiliary grid is the $m \times n$ grid with each undirected edge on any of the i paths be replaced by a directed edge in an opposite direction to the path. The search for an augmenting path starting from the super-source to super-sink can be performed by a breadth-first search (BFS) on the auxiliary grid. For example, Figure 1(a) shows the grid with 3 edge-disjoint paths. Figure 1(b) indicates an augmenting path for source node D on the auxiliary grid.

The new set of edge-disjoint paths for Figure 1(b) is presented in Figure 1(c), which contains one more edge-disjoint path than the original set shown in Figure 1(a). Thus, after each iteration the number of edge-disjoint paths can be increased by one. The algorithm terminates when no augmenting paths in the auxiliary grid remain, and at the same time, this algorithm also finds the maximum number d of source nodes that can be connected to the boundary. As each iteration might take $O(mn)$ time to search for an augmenting path, this algorithm takes $O(mnd)$ time.

In this paper, we present efficient and practical algorithms for edge-disjoint and vertex-disjoint reconfiguration problems based on the FORD-FULKERSON algorithm, determining the maximum number of disjoint paths. As the $m \times n$ grid has only N source nodes, we observe the sparity of source nodes in the grid and compress the grid by isolating source-node-free *rectangular blocks*. This compression reduces the size complexity of the grid from $O(mn)$ to a graph of size $O(\sqrt{mnN})$, i.e., from $O(N^2)$ to $O(N^{1.5})$ if m, n are $O(N)$, and retains all the properties of the original grid. This compression is optimal in the sense that the size of the graph cannot be further reduced, and it enables us to find an augmenting path in $O(\sqrt{mnN})$ time and solve the problem in $O(d\sqrt{mnN})$ time. The crux of our algorithm is to retain the connectivity information on the compressed graph and to find an augmenting path efficiently at each iteration.

In Section 2 we propose an optimal algorithm to isolate source-node-free rectangular blocks in a grid. Its routing through a source-node-free rectangular block will be discussed in Section 3. A necessary and sufficient condition for feasible routing, as well as the condition for reachability in a rectangular block, will be provided. Section 4 will discuss how a grid of $O(mn)$ size can be compressed into a graph of size $O(\sqrt{mnN})$ and how the reachability information can be retained in the compressed graph. Section 5 reviews the overall algorithm by discussing how an augmenting path be found in $O(\sqrt{mnN})$ time, how disjoint paths be identified in the original grid, and how the edge-disjoint paths problem be solved in $O(d\sqrt{mnN})$ time, i.e., $O(N^{2.5})$ time if d, m and n are $O(N)$. By tilting the grid by 45°, we show in Section 6 that the same technique can be modified and used to solve the vertex-disjoint problem. Section 7 concludes this paper.

2 Isolation of Rectangular Blocks.

Assume that the grid is $m \times n$ in size, where $m, n \leqslant N$. (Appendix A shows a simple $O(N)$ preprocessing step which reduces the size of the grid to $min(N, m) \times min(N, n)$ for *any* m, n.) A time-optimal $O(mn)$ algorithm to isolate source-node-free *rectangular blocks* in an $m \times n$ grid in an "optimal" fashion is presented. The *rectangular blocks* should be free of any source nodes, and totally disjoint from each other, i.e., their boundaries do not share any common grid vertices or edges (Figure 2). We shall show in the following subsection that the number of vertices and edges that are not covered by rectangular blocks is $\Theta(\sqrt{mnN})$ or $\Theta(N^{1.5})$ if m, n are $O(N)$, an asymptotically "optimal" isolation of rectangular blocks.

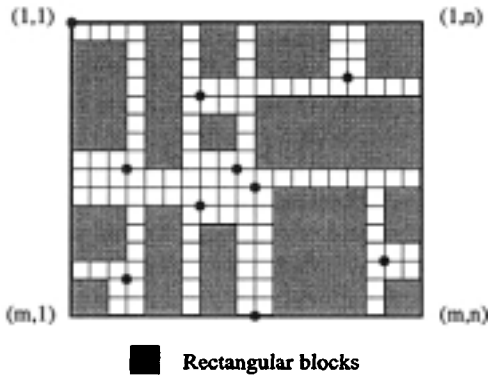


Figure 2: The isolation of rectangular blocks.

2.1 Algorithm for the Isolation of Rectangular Blocks. The isolation of the rectangular blocks in the grid G is done recursively by procedure RB-ISOLATION (Appendix B). Without loss of generality, assume $m \leqslant n$. The grid is then partitioned into 3 subgrids G_L, G_M, G_R of the same height (Figure 3), where (1) G_L, G_M, G_R represents the left, middle and right rectangular subgrids, (2) G_L and G_R individually share a common boundary with G_M, (3) G_M should cover the middle grid column and (4) G_M contains no source node except at its right and/or left boundaries. Note that G_L and G_R can be null while G_M always exists. If G_L or G_R exists, they will be handled recursively. If G_M is more than three grid columns wide, then G_M will become an isolated rectangular block after removing its leftmost and rightmost grid columns. Otherwise, no middle rectangular block can be isolated.

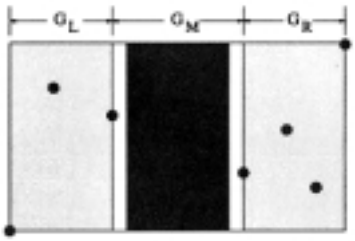


Figure 3: A step in Procedure RB-ISOLATION.

2.2 Performance Analysis of the Algorithm. Let $T(N, m, n)$ be the number of uncovered *cells* (a cell is the smallest square region in the grid), where N is the number of source nodes in an $m \times n$ grid, with $n \geqslant m$. Furthermore, let N_L and N_R be the number of source nodes in G_L, an $m_L \times n_L$ grid, and G_R, an $m_R \times n_R$ grid, respectively. According to the procedure RB-ISOLATION, we have the following conditions, $N_L + N_R = N, N_L \geqslant 0, N_R \geqslant 0, n_L \geqslant m_L, n_R \geqslant m_R$, and $m_L, m_R \leqslant \min(m, \lfloor \frac{n}{2} \rfloor)$, and $n_L, n_R \leqslant \max(m, \lfloor \frac{n}{2} \rfloor)$, and the recurrence,

(2.1)

$$T(N,m,n) \leqslant \begin{cases} 0 & \text{if } N = 0 \text{ or } m = 1, \\ T(N_L, m_L, n_L) + T(N_R, m_R, n_R) + 2m & \text{otherwise} \end{cases}$$

where the number of uncovered cells are at most $2m$ when G is partitioned into 3 subgrids. We can prove by induction on N that

$$T(N,m,n) \leqslant (k_1\sqrt{N} - k_2)\sqrt{mn} \tag{2.2}$$

for $N > 0$ and some constants $k_1, k_2 > 0$. Thus, from Eq. (2.2), the number of uncovered nodes and edges is $O(\sqrt{mnN})$, i.e., $O(N^{1.5})$ if m and n are $O(N)$.

2.3 Lower Bound of the Uncovered Cells. We shall show that the number of uncovered cells in an $m \times n$ grid with N source nodes is at least $\Omega(\sqrt{mnN})$ even when the source-node-free blocks to be isolated are not necessarily rectangular.

THEOREM 2.1. *As long as the blocks (free of source nodes) to be isolated do not contain any holes and $m \leqslant n \leqslant mN$, then the number of uncovered cells is at least $\Omega(\sqrt{mnN})$ for an $m \times n$ grid with N source nodes.*

Proof. (Sketch) Consider the N source nodes distributed evenly on an $m \times n$ grid as an $\sqrt{\frac{mN}{n}} \times \sqrt{\frac{nN}{m}}$ 2-dimensional array, the distance between any pair of source nodes, or between any source node and the grid boundary, will then be at least $\sqrt{\frac{mn}{N}}\left(1+\sqrt{\frac{m}{nN}}\right)^{-1}$ cells apart. The minimum weight spanning tree joining all N source nodes and the boundary must be of cost at least $\Omega(\sqrt{mnN})$.

3 Edge-Disjoint Paths in the Rectangular Blocks.

An augmenting path from the super-source to the super-sink (or a boundary vertex) can be found by a breadth-first search through the auxiliary grid. Since the rectangular blocks are free of any source nodes, the breadth-first search step does not have to consider all the grid vertices or edges in the rectangular block in finding an augmenting path as long as the reachability information between any pair of boundary grid vertices is known. A necessary and sufficient reachability condition on the boundary grid vertices of a rectangular block, and an asymptotically optimal algorithm to find the edge-disjoint paths in the rectangular block, will be given in the next two subsections.

3.1 Necessary and Sufficient Condition. Given a rectangular $p \times q$ block denoted by $[1 :: p] \times [1 :: q]$, vertices in the block can be represented by (i, j) where $1 \leqslant i \leqslant p$ and $1 \leqslant j \leqslant q$. The boundary of the block consists of the set of vertices $\mathcal{B} = \{(i,j) \mid i = 1,\ i = p,\ j = 1 \text{ or } j = q\}$. $\mathcal{C} = \{(1,1),(1,q),(p,1),(p,q)\}$ denotes the set of corner vertices of the block. $\mathcal{S}$ consists of all sources and $\mathcal{T}$ all sinks on the boundary of the grid. A set of edge-disjoint paths *satisfies* $\mathcal{S}$ and $\mathcal{T}$ if there are exactly $|\mathcal{S}| = |\mathcal{T}|$ edge-disjoint paths completely inside the block and each path is joining (or pairing up) a vertex in $\mathcal{S}$ with a vertex in $\mathcal{T}$.

A *cut* (*edge cut*) is defined as a set of block edges whose removal will partition the block into two components. In particular, *h-cut*(i) is a horizontal cut which denotes the ith row of block edges and *v-cut*(j) is a vertical cut which denotes the jth column of block edges. *h-cap*(i) and *v-cap*(i) is defined as the *capacity* (number of edges) of *h-cut*(i) and *v-cut*(j) respectively. The *demand of a cut* represents the least number of paths needed to pass through the cut in order to satisfy $\mathcal{S}$ and $\mathcal{T}$. In particular, *h-dem*(i) and *v-dem*(j) denote the least number of edge-disjoint paths needed to pass through the edges, from top to bottom, and from left to right, in *h-cut*(i) and *v-cut*(j) respectively. For instance, *h-dem*(i) or *v-dem*$(j) = k_1 - k_2$ where k_1 and k_2 are number of boundary vertices above the $(i+1)$st row or on the left of the $(j+1)$st column of the rectangular block and in $\mathcal{S}$ and $\mathcal{T}$ respectively . We say, a cut is *saturated* if its absolute value of demand equals its capacity and *overflowed* if its absolute value of demand is larger than its capacity. The conditions for the existence of edge-disjoint paths given in the following lemma is similar to those given in [9].

LEMMA 3.1. *Given a rectangular $p \times q$ block and two sets of boundary vertices $\mathcal{S}$ and $\mathcal{T}$ with $|\mathcal{S}| = |\mathcal{T}|$, there exists a set of edge-disjoint paths satisfying $\mathcal{S}$ and $\mathcal{T}$ if and only if all the h-cut(i) for $1 \leqslant i \leqslant p-1$ and v-cut(j) for $1 \leqslant j \leqslant q-1$ do not overflow.*

3.2 Paths Construction. Lemma 3.1 provides the condition which ensures the existence of a set of edge-disjoint paths satisfying $\mathcal{S}$ and $\mathcal{T}$. Such a set of edge-disjoint paths can be identified inside the rectangular grid by considering each block vertex as well as its adjacent edges one by one, row by row from top to bottom. When vertex (i, j) is considered, the flows (paths with direction) through its right edge $((i,j),(i,j+1))$ and its down edge $((i,j),(i+1,j))$ will be determined. Throughout the algorithm, we ensure that (1) the flow at each vertex is conserved, i.e., amount of inflow equals amount of outflow at each vertex, and (2) the new *h-cut*(i) and *v-cut*(j) in the remaining block do not overflow. This ensures that a set of edge-disjoint paths in the remaining block still exists.

THEOREM 3.1. *Given a rectangular $p \times q$ block and two sets of boundary vertices $\mathcal{S}$ and $\mathcal{T}$ with $|\mathcal{S}| = |\mathcal{T}|$, and all the h-cut$(i)$ for $1 \leqslant i \leqslant p-1$ and v-cut(j) for $1 \leqslant j \leqslant q-1$ not overflowed, a set of edge-disjoint paths in the grid satisfying $\mathcal{S}$ and $\mathcal{T}$ can be found in $O(pq)$ time.*

4 Construction of Reachability Graphs.

In this section, we shall show how the reachability information of an auxiliary grid in a rectangular block is retained, and how this information is used to accelerate the finding of the augmenting path. This reachability information is represented by a reachability graph R

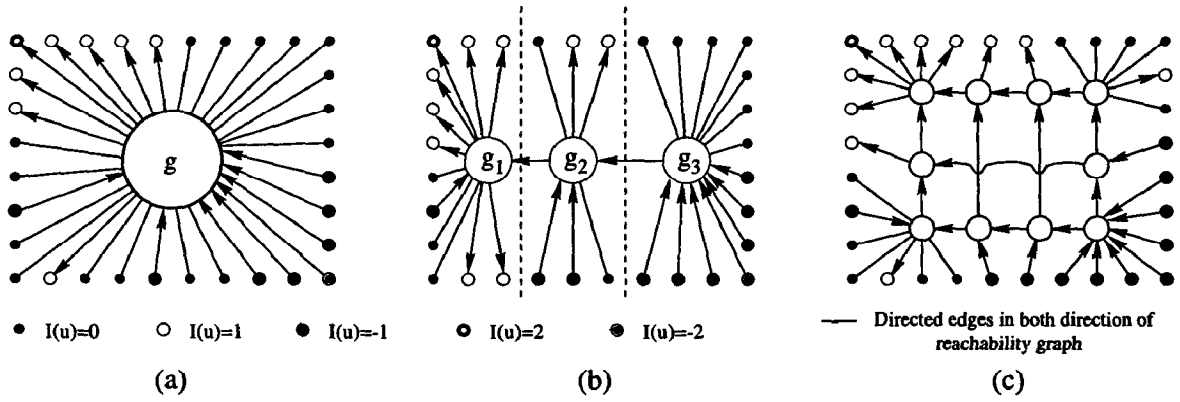


Figure 4: Reachability graphs for the Cases a, b and c.

which replaces the part of the auxiliary grid inside a rectangular block.

Define an inflow function $I(u)$, for $u \in \mathcal{B}$, to be the number of sources at u minus the number of sinks at u. In particular, $I(u)$ is *valid* if

$$I(u) = \begin{cases} -2,-1,0,1,2 & \text{if } u \in \mathcal{C} \\ -1,0,1 & \text{if } u \in \mathcal{B}-\mathcal{C} \end{cases} \text{ and } \sum_{u\in\mathcal{B}} I(u) = 0$$

In general, $I(u) > 0$ if $u \in \mathcal{S}$, < 0 if $u \in \mathcal{T}$ and otherwise $= 0$. Note that a set of edge-disjoint paths that satisfies $I(u)$ is equivalent to the set of edge-disjoint paths that satisfies $\mathcal{S}$ and $\mathcal{T}$. The inflow function $I(u)$ is called *solvable* if none of *h-cut*(i) for $1 \leqslant i \leqslant p-1$ and *v-cut*(j) for $1 \leqslant j \leqslant q-1$ are overflowed.

LEMMA 4.1. *Given any rectangular block with a valid and solvable inflow function $I(u)$, let a new valid inflow function $I'(u)$ be defined as*

$$I'(u) = \begin{cases} I(u)+1 & \text{if } u = x, \\ I(u)-1 & \text{if } u = y, \\ I(u) & \text{otherwise.} \end{cases} \quad \text{where } x, y \in \mathcal{B}$$

Then there exists an augmenting path from x to y in the auxiliary grid corresponding to the set of edge-disjoint paths satisfying $I(u)$ if and only if $I'(u)$ is solvable.

Based on the above lemma, we are able to determine the reachability information of each boundary vertex of a rectangular block to others based on $I(u)$. The instances of the inflow functions $I(u)$ are divided into 3 cases. In each case, we shall show how to construct the reachability graph R to represent the reachability of the boundary vertices of the rectangular block such that, for any two boundary vertices x and y, there is a path in R from x to y if and only if $I'(u)$ is valid and solvable.

Case a: Neither of the *h-cut*(i) and *v-cut*(j) is saturated. An augmenting path should exist between any pair of boundary vertices as long as $I'(u)$ remains valid. An example is given Figure 4(a).

Case b: Either *h-cut*(i) or *v-cut*(j) (not both) is saturated. The block is partitioned into components by its saturated cuts, and the reachability information of each component is similar to Case a and is represented by a star graph. An example of this case is given in Figure 4(b).

Case c: Both saturated *h-cut*(i) and *v-cut*(j) exist. The rectangular block is partitioned by the set of saturated cuts *h-cut*(i) and *v-cut*(j) into components. The reachability information of each of these *boundary* components will be represented by a star graph, in the same way as Cases a and b. An example is given in Figure 4(c).

It can be proved that the reachability information for all cases can be properly represented by a reachability graph of size $O(p+q)$.

5 Overall Algorithm.

Integrating the results in the previous sections, we can describe the overall algorithm as follows.

(1) Preprocess the grid (Appendix A).

(2) Isolate a set of source-node-free rectangular blocks in the grid (Section 2 and procedure RB-ISOLATION in Appendix B).

(3) Repeat

  (i) Based on the set of edge-disjoint paths found so far, construct the auxiliary grid and replace each rectangular block with its reachability graph (Section 4).

  (ii) Apply a breadth-first search to find an augmenting path from the super-source to the

super-sink (a source node to a boundary vertex) in the auxiliary grid. Exit if not found.

(iii) Update the edge-disjoint paths as well as the inflow function of the affected rectangular blocks.

(iv) Update the reachability graph of each affected rectangular block (Section 4).

(4) The set of edge-disjoint paths can be completed by constructing the paths which satisfies the inflow function in each rectangular block (Section 3 and Appendix C).

6 Vertex-Disjoint Paths in Grids.

The framework of our algorithm to solve the vertex-disjoint problem is similar to the one for the edge-disjoint paths problem, except that some steps need to be modified to cope with the vertex-disjoint requirement. The differences are: (1) the definition of rectangular blocks, (2) the construction of the paths inside the rectangular blocks, (3) the isolation of rectangular blocks in the grid, and (4) the construction of the reachability graph.

6.1 Tilted Rectangular Blocks. The type of rectangular block defined in the vertex-disjoint version is called a *tilted rectangular block*. The shape of the blocks remains rectangular but they are tilted at 45° to the horizontal grid line. A *tilted row* (*tilted column*) is a chain of vertices which form a straight line tilted at 45° clockwise (anti-clockwise) to the horizontal grid line.

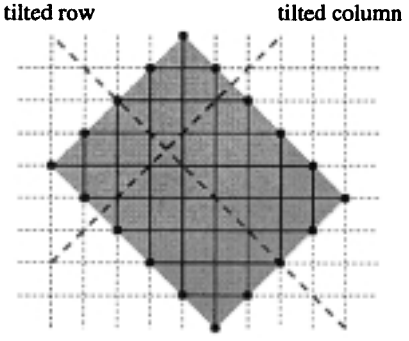


Figure 5: Formation of a tilted rectangular block.

For ease of visualizing the tilted rectangular block, imagine the tilted rectangular block is rotated 45° anti-clockwise so that the tilted rows and columns become horizontal and vertical respectively (Figure 6.1). The $h\text{-}cut(i)$ for $2 \leqslant i \leqslant p-1$ and $v\text{-}cut(j)$ for $2 \leqslant j \leqslant q-1$ are defined as *vertex cuts* which denote the ith tilted row and jth tilted column of vertices respectively in the tilted rectangular block. The capacity and demand of a cut are defined similarly. Note that when a cut has a boundary vertex in $\mathcal{S}$ or $\mathcal{T}$, the demand of this cut will take the worst-case value, i.e., the larger absolute value, by including or excluding the inflow/outflow at the boundary vertex. It can be proved that Lemma 3.1 still applies to the tilted rectangular block.

LEMMA 6.1. *Given a tilted rectangular $p \times q$ block and two sets of boundary vertices $\mathcal{S}$ and $\mathcal{T}$ with $|\mathcal{S}| = |\mathcal{T}|$, a set of vertex-disjoint paths exists which satisfies $\mathcal{S}$ and $\mathcal{T}$ if and only if all the $h\text{-}cut(i)$ for $2 \leqslant i \leqslant p-1$ and $v\text{-}cut(j)$ for $2 \leqslant j \leqslant q-1$ do not overflow.*

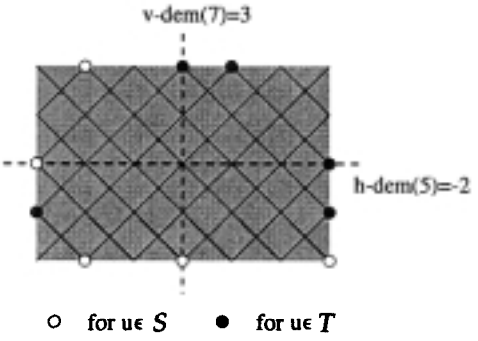


Figure 6: Tilted rectngular block rotated at 45° anti-clockwise.

Finally, the construction of vertex-disjoint paths satisfying $\mathcal{S}$ and $\mathcal{T}$ in a tilted rectangular block can also be done in a similar approach as the construction of the edge-disjoint paths.

6.2 Isolation of Rectangular Blocks. Since the definition of row and column in a tilted rectangular block has been changed, the isolation steps have to be modified in this vertex-disjoint case. A straightforward method is to transform the input $m \times n$ grid into an $(m+n-1) \times (m+n-1)$ tilted grid by padding the original grid with vertices on those tilted rows and columns which are not full. The padded grid is the smallest tilted grid enclosing the original grid. Applying the original isolation algorithm to the padded grid along the tilted rows and columns will result in $O((m+n)\sqrt{N})$ uncovered cells, i.e., $O(N^{1.5})$ if both m and n are $O(N)$. However, if m is much less than n, special handling methods are needed to obtain the optimal number of uncovered cells. The detailed description of the isolation algorithm will not be covered here.

6.3 Construction of Reachability Graphs. The construction of the reachability graph is the same as in the edge-disjoint case, except that the connection for the boundary vertex u to the center vertex is different

when u is on a saturated cut and $u \in \mathcal{S}$ or $\mathcal{T}$. The main difficulty lies in the fact that the cut consists of a set of vertices instead of edges and the two boundary vertices in a cut can be in $\mathcal{S}$ or $\mathcal{T}$. These boundary vertices have to be split into two vertices, one in each of the reachability graphs of two components.

7 Conclusion.

Routing problems have been studied widely by many researchers [8, 9, 11, 17]. Much of the research is on "specified" routing, that is, given a set of nets $\{(s_1, t_1), \ldots, (s_N, t_N)\}$ where (s_i, t_i) is a net such that $s_i \in \mathcal{S}$ and $t_i \in T$ with $1 \leqslant i \leqslant N$, the "specified" routing is to find n pairwise disjoint paths joining s_i and t_i where $1 \leqslant i \leqslant N$. In this paper, we have investigated a particular routing problem on a rectangular grid, which is called "unspecified" routing. The problem to be solved is finding n disjoint paths pairing vertices in $\mathcal{S}$ with vertices in $\mathcal{T}$, in the sense that vertices in $\mathcal{S}$ can be connected to any vertices in $\mathcal{T}$ as long as each vertex in $\mathcal{S}$ is connected to a distinct vertex in $\mathcal{T}$. This unspecified routing problem is a generalization of the reconfiguration problem or escape problem by defining $\mathcal{T}$ as the set of boundary vertices. It is easy to see that we can apply the technique described in this paper to solve this "unspecified" routing problem with the same $O(d\sqrt{mnN})$ or $O(N^{2.5})$ time complexity. Both the edge-disjoint and vertex-disjoint path algorithms presented in this paper are being implemented. Their experimental results will be presented in the full paper.

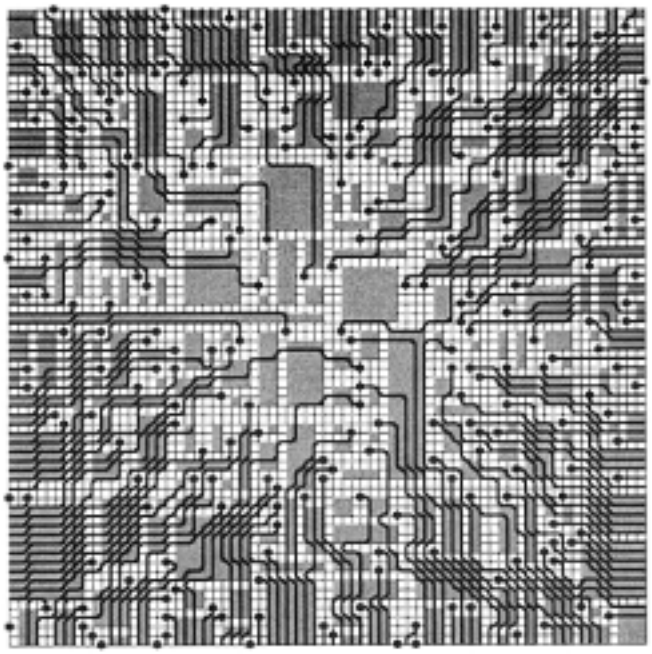


Figure 7: A set of edge-disjoint paths produced by the algorithm.

References

[1] Y. Birk and J. B. Lotspiech, *On finding non-intersecting straightline connections of grid points to the boundary*, J. of Algorithms, 13 (1992), pp. 636–656 (also appears in SODA 91').

[2] J. A. and U. S. R. Murty, *Graph Theory with Application*, North-Holland, Amsterdam, 1977.

[3] J. Bruck and V. P. Roychowdhury, *How to play bowling in parallel on the grid*, J. of Algorithms, 12 (1991), pp. 516–529.

[4] B. Codenotti and R. Tamassia, *A network flow approach to the reconfiguration of VLSI arrays*, IEEE Trans. on Computers, 40 (1991), pp. 118–121.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1991.

[6] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ.

[7] P. Klein, S. Rao, M. Rauch, and S. Subramanian, *Faster shortest-path algorithms for planar graphs*, Proceedings of the Twenty-Sixth Annual ACM Symp. on Theory of Computing, 1994, pp. 27–37.

[8] T. Lai and A. Sprague, *On the routability of a convex grid*, J. of Algorithms, 8 (1987), pp. 372–384.

[9] K. Mehlhorn and F. P. Preparata, *Routing through a rectangle*, JACM, 33 (1986), pp. 60–85.

[10] G. L. Miller and J. Naor, *Flow in planar graphs with multiple sources and sinks (extended abstract)*, Proc. of the 30th IEEE Symp. on Foundations of Computer Science, 1989, pp. 112–117.

[11] T. Nishizeki, N. Saito, and K. Suzuki, *A linear-time routing algorithm for convex grids*, IEEE Trans. on Computer-Aided Design, 4 (1985), pp. 68–75.

[12] L. Palios, *Connecting the maximum number of grid nodes to the boundary with non-intersecting line segments*, Proc. of the Scandinavian Workshop on Algorithm Theory, 1994, pp. 255–266.

[13] C. H. Papadimitrious and K. Steiglitz, *Combinatorial Optimization: Algorithm and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[14] H. Ripphausen-Lipa, D. Wagner, and K. Weihe, *The vertex-disjoint Menger problem in planar graphs*, Proc. of the Fourth Annual ACM-SIAM Symp. on Discrete Algorithms, 25–27 Jan. 1993, pp. 112–119.

[15] V. P. Roychowdhury and J. Bruck, *On finding non-intersecting paths in grids and its application in reconfiguring VLSI/WSI arrays*, Proc. of the First Annual ACM-SIAM Symp. on Discrete Algorithms, 1990, pp. 454-464.

[16] V. P. Roychowdhury, J. Bruck, and T. Kailath, *Efficient algorithms for reconfiguration in VLSI/WSI arrays*, IEEE Trans. on Comp., 39 (1990), pp. 480–489.

[17] A. Schrijver, *Finding k disjoint paths in a directed*

planar graph, SIAM J. on Computing, 23 (1994), pp. 780–788.

[18] H. Suzuki, T. Akama, and T. Nishizeki, *Finding Steiner forests in planar graphs*, Proc. of the First Annual ACM-SIAM Symp. on Discrete Algorithms, 1990, pp. 444–453.

Appendix A. Preprocessing of the Grid.

We have previously assumed that the grid sizes, m and n, are $O(N)$. However, this usually may not be the case. If the N source nodes are sparsely located in the grid, m and n can be arbitrarily large. In this section, we shall describe a preprocessing step which reduces the size of the grid to $min(N,m) \times min(N,n)$ by eliminating rows/columns such that the resultant grid would retain the same solution as the original grid. WLOG, let us first consider the removal of columns from the original grid. The removal of rows can be done similarly. Let $[b,e]$ denote a *cluster* of column i, $b \leqslant i \leqslant e$ and $p[b,e]$, the number of source nodes in $[b,e]$. We say $[b,e]$ is *routable* if and only if

$$\begin{array}{l} p[b,i] \leqslant 2(i-b+1) \\ \text{and } p[i,e] \leqslant 2(e-i+1) \end{array} \qquad \text{for all } i, b \leqslant i \leqslant e$$

Intuitively, the source nodes in a routable cluster $[b,e]$ can be *routed*, in other words, the existence of non-intersecting paths to the boundary nodes of the grid (in particular, the top and bottom boundaries) by only considering the columns in $[b,e]$. The two inequalities ensure that there are enough columns on both sides of the cluster even when all the source nodes are routed towards the same side. For example in Figure 8, a routable cluster would include at least columns 3 to 8, [2,9], [2,8], [3,10], $[x,y]$ with $x \leqslant 3$ and $y \geqslant 8$ are all routable.

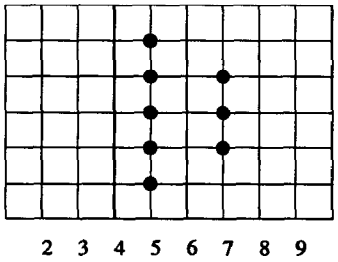


Figure 8: Example of a routable cluster.

LEMMA A.1. *All the source nodes in a routable cluster that can be routed to the boundary, can also be routed to the top and bottom boundary nodes of the cluster.*

Proof. Consider a routable cluster $[c_1,c_2]$. For all the source nodes that can be routed to the boundary, we have a set of paths connecting them to the destination nodes on the four boundaries of $[c_1,c_2]$. Let us consider a solution with the minimum number of horizontal edges, we claim that there will be no more than 2 destination nodes on each of the columns c_1, c_2 connecting to sources nodes in $[c_1,c_2]$, i.e. all the source nodes can be routed to the top/bottom boundary of $[c_1,c_2]$. WLOG, let us assume the contrary that c_2 contains more than 2 destination nodes, consider the column k for the maximum k, such that $k < c_2$ and contains less than 2 destination nodes. The routable property implies that

$$p[k+1,c_2] \leqslant 2(c_2-k)$$

So there is at least one source node located in cluster $[c_1,k]$ but with its destination node in cluster $[k+1,c_2]$. This implies at least one path passing through column k from cluster $[c_1,k]$ to $[k+1,c_2]$. Therefore, we can direct this path to a new destination node on column k and reduce the number of horizontal edges (supposed to be minimum). If c_1 contains more than 2 destination nodes, we will get a contradiction similarly.

In the following, we shall describe an $O(N)$ algorithm to find a set of disjoint routable clusters of total size $O(N)$ to cover all N source nodes in the grid.

A cluster $[b,e]$ is called *right routable* if for all i, $b \leqslant i \leqslant e$, $p[i,e] \leqslant 2(e-i+1)$. Similarly, *left routable* if $p[b,i] \leqslant 2(i-b+1)$ for all i, $b \leqslant i \leqslant e$.

The basic technique to find a set of disjoint right or left routable clusters is by scanning. WLOG, the algorithm to find the right routable clusters starts with the left most column, i.e., smallest indexed column, say column b containing source node(s), st each column to its right one by one until the first column $e > b$ such that $p[b,e] \leqslant 2(e-b+1)$. We shall prove that $[b,e]$ is the first right routable cluster in the set. The algorithm repeats itself to find the other right routable clusters by starting at the first column to the left of $[b,e]$ and containing source node. Obviously, the algorithm takes linear time and can find a set of disjoint right routable clusters to cover all the source nodes. Similarly, the left routable clusters can be found by scanning from the right most column.

LEMMA A.2. *$[b,e]$ is right routable.*

Proof. For any $b \leqslant i \leqslant e$, $p[i,e] = p[b,e] - p[b,i-1] < 2(e-b+1) - 2(i-b) = 2(e-i+1)$ as $p[b,i-1] > 2(i-1-b+1) = 2(i-b)$ (from the algorithm).

LEMMA A.3. *Assume $\{[b_i,e_i] \mid 1 \leqslant i \leqslant k\}$ is a set of disjoint right routable clusters, then $[x,y]$ is also a right routable cluster where $e_j \leqslant y < b_{j+1}$ for all $1 \leqslant j \leqslant k$ and $x \leqslant y$.*

Proof. We claim that for $x \leqslant z \leqslant y$, the number of source nodes $p[z,y] \leqslant 2(y-z+1)$.

1. If $e_{j'} < z < b_{j'+1}$ for some j', the result follows.
2. If $b_{j'} \leqslant z \leqslant e_{j'}$ for some j', since $[b_{j'}, e_{j'}]$ is a right routable cluster, $p[z,e_{j'}] \leqslant 2(e_{j'}-z+1)$. Therefore, $p[z,y] \leqslant 2(y-z+1)$.

LEMMA A.4. *Assume $\{[b'_i,e'_i] \mid 1 \leqslant i \leqslant k\}$ is a set of disjoint left routable clusters, then $[x,y]$ is also a left routable cluster where $e'_j < x \leqslant b'_{j+1}$ for all $1 \leqslant j \leqslant k$ and $x \leqslant y$.*

Proof. The proof is similar as Lemma A.3.

We define the operation *union*, $\cup$, of two cluster $[b_1,e_1]$ and $[b_2,e_2]$ where $b_1 \leqslant b_2$ as

$$[b_1,e_1] \cup [b_2,e_2] = \begin{cases} [b_1,e_1],[b_2,e_2] & \text{if } e_1 < b_2 \\ [b_1,e_2] & \text{if } b_2 \leqslant e_1 < e_2 \\ [b_1,e_1] & \text{if } e_2 \leqslant e_1 \end{cases}$$

For two sets of disjoint cluster C_1 and C_2, we define the operation union $\cup$ as

$$C_1 \cup C_2 = \bigcup [b,e] \quad \text{where } [b,e] \in C_1 \text{ or } C_2$$

Let $RRC = \{[b_i,e_i]\}$ be the set of disjoint right routable cluster and $LRC = \{[b'_i,e'_i]\}$ be the set of disjoint left routable cluster. We can construct another set of disjoint routable cluster $RC = RRC \cup LRC$.

LEMMA A.5. *RC is a set of disjoint routable cluster.*

Proof. Let $RC = \{[b,e]\}$. By the definition of union, we have $b = b_i$ or b'_i for some i and $e = e_j$ or e'_j for some j Moreover, $e'_k < b_i \leqslant b'_{k+1}$ for all i, otherwise RC is not disjoint. Similarly, $e_k \leqslant e'_i < b_{k+1}$ for all i. Then, we have $e'_i < b \leqslant b'_{i+1}$ for some i and $e_j \leqslant e < b'_{j+1}$ for some j. By Lemma A.3 & A.4, $[b,e]$ is right and left routable, and hence it is routable. Therefore, RC is a set of disjoint routable cluster.

LEMMA A.6. *The size of RC is at most N.*

Proof. Consider a cluster $[b,e] \in RC$, assume it contains the clusters $\{[b_k,e_k] \mid i \leqslant k \leqslant j\} \subseteq RRC$ and $\{[b_{k'},e_{k'}] \mid i' \leqslant k' \leqslant j'\} \subseteq LRC$. By the algorithm, $p[b_k,e_k] > 2(e_k - b_k)$. Then we have $e_k - b_k + 1 \leqslant \frac{p[b_k,e_k]+1}{2}$. Moreover, since this cluster contains at least one source node, that column containing source node will be covered by a cluster in $\{[b_{k'},e_{k'}]\}$. Similarly, we have $e_{k'} - b_{k'} + 1 \leqslant \frac{p[b_{k'}-e_{k'}]+1}{2}$. This cluster $[b_{k'},e_{k'}]$ contains at least one column covered by a cluster in $\{[b_k,e_k]\}$. Conclusively, we have

$$\begin{aligned} e-b+1 &\leqslant \sum_{k=i}^{j}(e_k-b_k+1) + \sum_{k=i'}^{j'}(e_{k'}-b_{k'}+1) \\ &\quad - \frac{j-i+1}{2} - \frac{j'-i'+1}{2} \\ &\leqslant \sum_{k=i}^{j}(\frac{p[b_k,e_k]+1}{2}) + \sum_{k=i'}^{j'}(\frac{p[b_{k'},e_{k'}]+1}{2}) \\ &\quad - \frac{j-i+1}{2} - \frac{j'-i'+1}{2} \\ &\leqslant p[b,e] \end{aligned}$$

The size of the set of disjoint routable cluster RC is $\sum_{[b,e]\in RC}(e-b+1)$ which is less than $\sum_{[b,e]\in RC} p[b,e]$, i.e., N.

Appendix B. Procedure RB-ISOLATION(G).

Without loss of generality, let G be an $m \times n$ grid with $n \geqslant m$.

(1) (Base case) If the number of source vertices N is 0 or $m = 1$, return.

(2) Let grid column i, where $1 \leqslant i \leqslant \lfloor \frac{n}{2} \rfloor$, be the minimum column such that all columns from $i+1$ to $\lfloor \frac{n}{2} \rfloor$ are free of source nodes. Column i is then the common boundary between G_L and G_M.

(3) Let grid column j, where $\lfloor \frac{n}{2} \rfloor < j \leqslant n$, be the maximum column such that all columns from $\lfloor \frac{n}{2} \rfloor + 1$ to $j-1$ are free of source nodes. Column j is then the common boundary between G_M and G_R.

(4) If $j \geqslant i+3$, then the subgrid containing grid columns from $i+1$ to $j-1$ will be an isolated rectangular block in G_M. (Otherwise, we do not have an isolated rectangular block in G_M.)

(5) If $i \geqslant 2$, then G_L, the subgrid containing grid columns from 1 to i, will be handled recursively, i.e., RB-ISOLATION(G_L).

(6) If $j \leqslant n-1$, G_R, the subgrid containing grid columns from j to n, will be handled recursively, i.e., RB-ISOLATION(G_R).

Appendix C. Algorithm for Edge-Disjoint Paths Construction in Rectangular Blocks.

In this section, we exhaust the cases for the assignment of flow $f(b)$ and $f(c)$ at vertex (i,j) (Inside the bracket

are the changes of demands of $h\text{-}cut(i)$ and $v\text{-}cut(j)$ due to the assignment of flows $f(b)$ and $f(c)$, at the same time, the capacity $h\text{-}cap(i)$ and $v\text{-}cap(j)$ will be decremented by one):

1. $f(a)+f(d_j)=2$: $f(b)=f(c)=1$ ($v\text{-}dem'(j)=v\text{-}dem(j)-1$ and $h\text{-}dem'(i)=h\text{-}dem(i)-1$).
2. $f(a)+f(d_j)=-2$: $f(b)=f(c)=-1$ ($v\text{-}dem'(j)=v\text{-}dem(j)+1$ and $h\text{-}dem'(i)=h\text{-}dem(i)+1$).
3. $f(a)+f(d_j)=0$:
   (a) $v\text{-}dem(j)=vc(j)$ or $h\text{-}dem(i)=-hc(i)$: $f(b)=-1$ and $f(c)=1$ ($v\text{-}dem'(j)=v\text{-}dem(j)-1$ and $h\text{-}dem'(i)=h\text{-}dem(i)+1$).
   (b) $v\text{-}dem(j)=-vc(j)$ or $h\text{-}dem(i)=hc(i)$: $f(b)=1$ and $f(c)=-1$ ($v\text{-}dem'(j)=v\text{-}dem(j)+1$ and $h\text{-}dem'(i)=h\text{-}dem(i)-1$).
   (c) Otherwise: $f(b)=f(c)=0$ ($v\text{-}dem'(j)=v\text{-}dem(j)$ and $h\text{-}dem'(i)=h\text{-}dem(i)$).
4. $f(a)+f(d_j)=1$:
   (a) $v\text{-}dem(j)=vc(j)$ or $h\text{-}dem(i)=-hc(i)+1$: $f(b)=0$ and $f(c)=1$ ($v\text{-}dem'(j)=v\text{-}dem(j)-1$ and $h\text{-}dem'(i)=h\text{-}dem(i)$).
   (b) Otherwise: $f(b)=1$ and $f(c)=0$ ($v\text{-}dem'(j)=v\text{-}dem(j)$ and $h\text{-}dem'(i)=h\text{-}dem(i)-1$).
5. $f(a)+f(d_j)=-1$:
   (a) $v\text{-}dem(j)=-vc(j)$ or $h\text{-}dem(i)=hc(i)-1$: $f(b)=0$ and $f(c)=-1$ ($v\text{-}dem'(j)=v\text{-}dem(j)+1$ and $h\text{-}dem'(i)=h\text{-}dem(i)$).
   (b) Otherwise: $f(b)=-1$ and $f(c)=0$ ($v\text{-}dem'(j)=v\text{-}dem(j)$ and $h\text{-}dem'(i)=h\text{-}dem(i)+1$).

Proof. Correctness of the algorithm

We shall prove the correctness of the algorithm by induction on i, for $1 \leqslant i \leqslant p$. Hypothesis: after we consider the vertex (i,j), (1) the new cuts $h\text{-}cut'(i)$ and $v\text{-}cut'(j)$ in the remaining block are not overflowed, i.e., $|h\text{-}dem'(i)| \leqslant h\text{-}cap'(i)$ and $|v\text{-}dem'(j)| \leqslant v\text{-}cap'(j)$ and (2) the flow at vertex (i,j) is conserved.

Assume it holds for the vertices $(i,1),\ldots,(i,j-1)$. Let us consider vertex (i,j),

1. $f(a)+f(d_j)=2$:
   Since $v\text{-}dem(j) \geqslant v\text{-}dem'(j-1)+1 \geqslant -p+i+1$ and $v\text{-}dem(j) \leqslant p-i+1$, we have $|v\text{-}dem(j)-1| \leqslant p-i$, i.e., $|v\text{-}dem'(j)| \leqslant v\text{-}cap'(j)$. Similarly, $|h\text{-}dem'(i)| \leqslant h\text{-}cap'(i)$.
2. $f(a)+f(d_j)=-2$: Similar to previous case.
3. $f(a)+f(d_j)=0$:
   (a) If $v\text{-}dem(j)=v\text{-}cap(j)$: We claim that $h\text{-}dem(i) \leqslant q-j-1$. If $h\text{-}dem(i) \geqslant q-j$, we have $h\text{-}dem(i)+v\text{-}dem(j) \geqslant p+q-i-j+1$. That means the net number of boundary vertices in $\mathcal{S}$ to the left of $v\text{-}cut(i)$ or on top of $h\text{-}cut(j)$ is greater than $p+q-i-j$. However, the number of boundary vertices to the right of $v\text{-}cut(j)$ and on top of $h\text{-}cut(j)$ is no greater than $p+q-i-j$. It makes a contradiction if the claim is not true. The other conditions follow directly in the algorithm. Then we have $|v\text{-}dem'(j)| \leqslant v\text{-}cap'(j)$ and $|h\text{-}dem'(i)| \leqslant h\text{-}cap'(i)$.
   The case for $h\text{-}dem(j)=-h\text{-}cap(j)$ can be proved in similar way.
   (b) The cases for $v\text{-}dem(j)=-v\text{-}cap(j)$ or $h\text{-}dem(i)=h\text{-}cap(i)$ can be proved in similar way.
   (c) Otherwise: Trivial.
4. Cases when $f(a)+f(d_j)=1$ or -1 can be proved in similar way.

The conservation of flow at vertex (i,j) follows from the algorithm directly. It is also easy to see that the assignment of flow function at each edge takes constant time and thus the whole algorithm takes $O(pq)$ time.

Assume all the vertices in the ith row have been considered where $1 \leqslant i \leqslant p$, the flow patterns of those edges $\{((i,1),(i+1,1)),((i,2),(i+1,2)),\ldots,((i,q),(i+1,q))\}$ in the horizontal cut $h\text{-}cut(i)$ for the $p \times q$ block have been determined and can be treated as the new sources and sinks for the remaining $(p-i) \times q$ block denoted by $[i+1::p] \times [1::q]$.

Once the flow patterns of all block edges are determined, we can follow the flow direction of each block edge to construct the set of edges-disjoint paths satisfying the sets $\mathcal{S}$ and $\mathcal{T}$. The edge-disjoint path corresponding to each flow pattern at each block vertex is as follows:

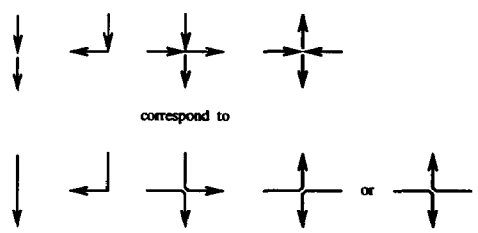


Figure 9: Formation of Paths.